Middleware de Integração entre o Ambiente AulaNet e o Ginga

Hugo Fuks[1], Marco Aurélio Gerosa[2], Celso Gomes Barreto[1] e Carlos José Pereira de Lucena[1]

[1]Departamento de Informática – PUC-Rio
Rua Marquês de São Vicente, 225 RDC – Gávea
22453-900 – Rio de Janeiro – RJ – Brasil

[2]Centro Universitário Vila Velha
Rua Comissário José Dantas de Melo, 21 – Boa Vista
29102-770 – Vila Velha – ES – Brasil

hugo@inf.puc-rio.br, gerosa@les.inf.puc-rio.br,
celso@les.inf.puc-rio.br e lucena@inf.puc-rio.br

Resumo

O middleware Ginga possibilita o desenvolvimento de aplicações interativas para a TV Digital de forma independente da plataforma de hardware dos fabricantes de terminais de acesso. Neste artigo é proposto um middleware de integração entre o Ambiente AulaNet e o Ginga, de modo a prover uma plataforma para o ensino-aprendizagem através da TV digital. O AulaNet é um ambiente gratuito de ensino-aprendizagem online. O middleware será desenvolvido utilizando técnicas de desenvolvimento baseado em componentes, aproveitando da nova arquitetura do ambiente.

Abstract

The Ginga middleware allows the development of interactive applications to the digital TV independently from the hardware platform of the set-top box providers. In this paper a middleware is proposed to integrate the AulaNet environment with Ginga, in a way to provide a platform for teaching-learning through the digital TV. The middleware will be developed using component based development techniques, taking into account the existing system architecture.

1. Introdução

A Educação é uma necessidade do Brasil, que ocupa a 72ª posição no ranking mundial de educação. Para melhorar este quadro, é necessário investir e variar as modalidades de ensino-aprendizagem. A TV digital oferece uma perspectiva promissora. A TV digital é um aparelho em tese mais simples de operar que o microcomputador e com maior potencial de disseminação. O decreto presidencial No 5.820, de 29 de junho de 2006, estabelece um prazo de 10 anos para que toda transmissão terrestre do país seja digital. Em relatório da Pesquisa Nacional por Amostra de Domicílio (PNAD) de 2005,

elaborado pelo IBGE, aponta-se 13,7% dos domicílios com computador com acesso à internet e 91,4% com aparelhos de TV.

Diversas pesquisas exploram as possibilidades de interação providas pela TV digital na Educação [Santos et al., 2005][Oliveira & Albuquerque, 2006][Amaral et al, 2004][Sancrini, 2005][Waisman, 2002]. A própria ANATEL e outros órgãos governamentais reconhecem o potencial da TV digital para educação a distância [Tome et al., 2001][CPqD, 2005]. O MEC lançou o programa TV Digital Interativa, visando levar às escolas conteúdos desenvolvidos para a tecnologia da TV digital. O BNDES criou o Programa de Apoio à Implementação do Sistema Brasileiro de TV Digital Terrestre (Protvd), com orçamento de R$ 1 bilhão e vigência até 31 de dezembro de 2013.

A TV digital favorece a integração de várias mídias na Educação. Apesar dos jovens já estarem em constante contato com esta pluralidade, na sala de aula utiliza-se geralmente a fala e a escrita como formas de comunicação e publicação de conhecimento. A capacidade multimídia da TV digital possibilita o uso de recursos mais atraentes para esta nova geração de alunos. Os alunos, ao produzirem conteúdo, tornam-se mais capazes de lidar com as novas modalidades de comunicação e expressão, agindo como *camera man*, desenhista, ator, redator, entre outros. Promove-se a educação para as mídias e com as mídias [Belloni, 2001].

Com a capacidade de processamento do *set-top box* é possível desenvolver sistemas para aprofundar os conhecimentos adquiridos através de testes, jogos educativos e outros recursos didáticos oferecidos pela tecnologia. A futura capacidade de interação possibilitará um contato entre estudantes e professores para troca de experiências e aprofundamento e fixação do conhecimento. Para gerenciar os recursos de interação e os conteúdos didáticos (objetos de aprendizagem), é necessário um ambiente, comumente chamado de LMS (*Learning Management System*).

O Ambiente AulaNet é um LMS gratuito voltado para o ensino-aprendizagem online. O AulaNet é desenvolvido em conjunto pela PUC-Rio e pela empresa EduWeb-SGPS desde 1997 e está disponível em português, inglês e espanhol. O Ambiente AulaNet é utilizado em diversas instituições de ensino e em empresas para treinamento de seus funcionários. Alguns clientes da plataforma são CCAD/PUC-Rio (6.000 alunos), EMBRAER (17.000), TV Globo (8.000), SENAC (6.000), Nextel (3.000), Sest-Senat (2.800), ALL, Ultragás, Alpargatas, Polícia Civil do Rio de Janeiro, Vivo, ABERJ, EletroPaulo, IBDE, Nextel, TCE do Rio de Janeiro, SEST/SENAT, etc. Algumas universidades que usam o AulaNet são PUC-Rio, Federal do Rio de Janeiro, Federal da Bahia, Federal de Minas Gerais, Federal de Mato Grosso, Católica de Salvador, Faculdade Michelangelo (Brasília), Faculdades Integradas de Boituva (SP), Universidade Católica de Petrópolis, Universidade de Macapá, Universidade Tecnológica do Panamá, Universidade de Aveiro (Portugal), Instituto Politécnico de Gaya (Portugal), Universidade da Madeira (Portugal).

Num consórcio de pesquisa de instituições brasileiras foi definido e implementado o middleware Ginga, que possibilita o desenvolvimento de aplicações interativas para a TV Digital de forma independente da plataforma de hardware dos fabricantes de terminais de acesso (*set-top boxes*). Neste artigo é proposto um middleware de integração entre o Ambiente AulaNet e o Ginga, de modo a prover uma plataforma para o ensino-aprendizagem através da TV digital. O uso de um ambiente gerenciador de aprendizagem gratuito facilitará a experimentação e a pesquisa sobre o uso a TV digital na Educação, possibilitando que os pesquisadores de diversas instituições tenham uma plataforma de teste e aplicação para suas pesquisas. A nova versão do Ambiente

AulaNet está sendo desenvolvida com base em uma arquitetura baseada em componentes, o que facilitará esta integração.

2. Desenvolvimento Baseado em Componentes

O Desenvolvimento Baseado em Componentes se mostra atraente em situações onde há necessidade de recompor a aplicação devido a particularidades de hardware e software, como é o caso da TV Digital. Nesta seção são discutidos alguns conceitos, benefícios e dificuldades desta tecnologia.

Componentes de software são substituíveis, reusáveis e interoperáveis e utilizados para compor a aplicação final [Gimenes & Huzita, 2005]. Um componente de software é [D'Souza & Wills, 1998]: Um pacote coerente de software que (a) pode ser desenvolvido e instalado independentemente como uma unidade, (b) tem interfaces explícitas e bem definidas para os serviços que provê, (c) tem interfaces explícitas e bem definidas para os serviços que espera de outros, e (d) pode ser utilizado para composição com outros componentes, sem alterações em sua implementação, podendo eventualmente ser customizado em algumas de suas propriedades.

A interface é o contrato de utilização do componente [Szyperski, 1997]. Respeitando-se os contratos, pode-se alterar a implementação interna do componente ou substituí-lo por outro, sem modificar seus clientes. A interface define as maneiras de utilizar o componente, separando a especificação da implementação. Um componente apresenta múltiplas interfaces correspondendo aos conjuntos de serviços que visam diferentes necessidades dos clientes [D'Souza & Wills, 1998]. Normalmente, o componente possui pelo menos uma interface relativa aos serviços disponibilizados (interface de negócio) e outra à conexão com a infra-estrutura de execução (interface de sistema), onde são tratados serviços técnicos, como os relacionados ao ciclo de vida, à instalação e à persistência. As interfaces são classificadas em fornecidas (*provided interfaces*) e requeridas (*required interfaces*) [Councill & Heineman, 2001].

O desenvolvimento baseado em componentes considera pelo menos duas visões da arquitetura: arquitetura de aplicação e arquitetura técnica [D'Souza & Wills, 1998]. Na arquitetura de aplicação são definidas a função de cada componente no contexto do sistema e a interação entre eles, de forma independente da tecnologia de suporte. A arquitetura técnica trata a tecnologia de suporte, independentemente do domínio da aplicação. Um componente de software é instalado em uma plataforma de execução e segue um modelo de componentes [Szyperski, 1997]. É concebido para ser autocontido, reusável e substituível e prover serviços específicos de uma maneira coesa e bem definida.

Uma das principais motivações para se desenvolver um software baseado em componentes é sua manutenção. Os componentes são substituíveis para atualização ou correção, muitas vezes sem precisar alterar ou recompilar a aplicação como um todo [D'Souza & Wills, 1998]. Componentes são adicionados, removidos ou substituídos por versões mais robustas ou mais apropriadas ao hardware, ao sistema operacional ou aos produtos legados com os quais o sistema tenha que operar. A modularidade obtida com a componentização facilita a localização do código a ser alterado e o encapsulamento da alteração. Dada a presente imaturidade da tecnologia da TV digital esta característica se mostra bastante apropriada.

O reuso também é freqüentemente citado como um benefício da componentização. O reuso favorece a redução dos esforços de desenvolvimento e a qualidade do produto final, por colocar em uso código já utilizado e testado em outras situações [Krueger,

1992]. Blocos com granularidade baixa (como uma classe) e alta (como um sistema) são difíceis de reusar, pois são genéricos ou específicos demais. O desenvolvimento baseado em componentes trabalha com blocos com granularidade média, mais propícia para o reuso em linhas de produtos. Isto possibilitará compor diferentes versões do ambiente AulaNet para os diferentes públicos alvos e tecnologias. O software baseado em componentes torna-se mais adaptável e extensível a diversas situações [D'Souza & Wills, 1998].

Outra vantagem da componentização é o encapsulamento de conhecimento e uma programação de alto nível. O desenvolvimento do AulaNet, por lidar com a aprendizagem colaborativa, envolve diversas áreas de conhecimento, como informática, psicologia, pedagogia, sociologia e cognição. Ao se integrar à TV digital, envolve várias complexidades técnicas inerentes a tecnologia e a sistemas distribuídos e multi-usuário. Um desenvolvedor não precisa conhecer os detalhes de implementação dos componentes para utilizá-los para compor as aplicações. Quem integra componentes se especializa nesta atividade abstraindo os detalhes de implementação, tendo uma visão mais abrangente e mais próxima do domínio de aplicação.

No desenvolvimento de sistemas colaborativos voltados para aprendizagem, os requisitos raramente são claros o suficiente para possibilitar uma especificação precisa antecipada do comportamento do sistema [Gutwin & Greenberg, 2000]. Por envolver um grupo, multiplicam-se as possibilidades de interações e aumenta a demanda por sincronismo e concorrência de acesso, o que dificulta a construção de mecanismos de interação adequados. A componentização facilita a prototipação, pois o sistema é recomposto para experimentar idéias. Esta capacidade é especialmente útil em sistemas com requisitos não definidos e instáveis, como é o caso de groupware. O desenvolvedor experimenta e prototipa diversas configurações iterativamente para refinar os requisitos do sistema e o suporte à colaboração. A prototipação e o desenvolvimento iterativo possibilitam colocar o sistema em produção mais cedo, de modo a refinar gradualmente os requisitos e construir o sistema com base no aprendizado obtido com a realimentação [Teles, 2004].

A decomposição do sistema possibilita a definição de componentes independentes, que podem ser subcontratados ou alocados para outras equipes, o que favorece o desenvolvimento paralelo e em grupo. Em alguns sistemas [Won et al., 2005; Slagter & Biemans, 2000; Li & Muntz, 1998; Hummes & Merialdo, 2000], os próprios usuários finais recompõem e re-configuram os componentes, adaptando a aplicação para suas necessidades específicas. A aplicação é incrementada para acompanhar as características das tarefas e para prototipar e experimentar configurações.

Software é evolutivo, e groupware é evolutivo por natureza [Tam & Greenberg, 2004]. A composição e as características dos grupos de trabalho se alteram ao longo do tempo, assim como as tarefas executadas. O grupo aprende, surgem afinidades e conflitos entre os membros, entram e saem pessoas, levando o grupo a mudar continuamente. A componentização provê a capacidade de montar e evoluir um ambiente de trabalho específico, selecionando e configurando um conjunto de ferramentas colaborativas específicas para suas necessidades.

O desenvolvedor de um componente é beneficiado pela infra-estrutura de execução, que provê serviços básicos como persistência, interconexão, escalabilidade, etc., eliminando a necessidade de implementar estes serviços. Algumas infra-estruturas possibilitam a integração de forma transparente para o programador de componentes desenvolvidos e disponibilizados em diferentes linguagens de programação, tecnologias e plataformas [D'Souza & Wills, 1998].

Com relação às dificuldades, o desenvolvimento baseado em componentes demanda um esforço inicial maior de projeto e implementação para montar a infra-estrutura do sistema e construir uma biblioteca robusta de componentes reusáveis. Projetar e preparar um pedaço de software para futuro reuso aumenta a necessidade de flexibilidade, documentação, estabilidade e abrangência do software [Moore & Bailin, 1991]. O software deve ser bem documentado, testado e deve ter um esquema robusto de validação [Pfleeger, 2001]. Os componentes não podem ser nem muito genéricos e nem muito específicos [Oliveira, 2001].

O custo do estudo e entendimento de como usar e instalar os componentes também é outra dificuldade associada com o reuso. Às vezes é mais rápido desenvolver um componente do que procurar por um pronto, estudá-lo e adaptá-lo [Pfleeger, 2001]. A menos que o custo de aprendizagem seja amortizado por vários projetos ou que o ganho de produtividade e qualidade sejam expressivos, o investimento inicial não se torna atraente.

3. O Ambiente AulaNet

No AulaNet é disponibilizado um conjunto de serviços e o coordenador seleciona os que serão utilizados em seu curso e configura-os de acordo com as dinâmicas educacionais que serão adotadas nas turmas. Nas suas primeiras versões, os serviços do AulaNet eram classificados em serviços *administrativos*, de *avaliação* e *didáticos*, que é uma abordagem comum em ferramentas educacionais [Edutools, 2005]. Entretanto, esta abordagem levou os docentes que usavam o ambiente a ensinar da maneira vertical tradicional: professando informações com pouca interação entre eles e os aprendizes, e sem interação entre os aprendizes. Contudo, o que se espera de um aprendiz na colaboração é um alto grau de interação com seus colegas e com os docentes, que por sua vez devem agir como mediadores e coordenadores ao invés de entregadores de informação. Os serviços do AulaNet foram reorganizados com base no modelo 3C de colaboração, para incentivar a colaboração [Fuks, 2000].

Figura 1. Posicionamento dos serviços do AulaNet no triângulo apresentado por Borghoff & Schlichter [2000]

Os serviços de colaboração do ambiente AulaNet são atualmente organizados em serviços de comunicação, de coordenação e de cooperação. A Figura 1 ilustra o posicionamento dos serviços do AulaNet no triângulo apresentado em [Borghoff &

Schlichter, 2000]. Os serviços do AulaNet estão posicionados na parte externa do triângulo.

No modelo 3C, a colaboração é vista a partir da comunicação, coordenação e cooperação [Ellis et al, 1991]. Ao enxergar o problema sob a perspectiva do modelo 3C e utilizar a componentização organizada em função deste modelo, as alterações na colaboração são mapeadas ao suporte computacional, que é substituído ou acrescentado na medida da necessidade. Os componentes possibilitam lidar com o projeto da colaboração em um alto nível. Através da componentização, o Ambiente AulaNet se torna mais adaptável à variedade de grupos que o utilizam, que neste caso são compostos por alunos e professores de diferentes regiões do país e do mundo, e às diversas características dos cursos aplicados através dele, que variam no conteúdo (desde cursos para crianças até cursos de pós-graduação) e na abordagem pedagógica utilizada.

Na abordagem utilizada no desenvolvimento da nova versão do Ambiente AulaNet, o sistema colaborativo é estruturado em componentes que encapsulam as dificuldades técnicas de sistemas distribuídos e multi-usuário e refletem os conceitos da colaboração, modelados pelo modelo 3C. Este ferramental instrumenta o desenvolvimento da camada de negócio da aplicação, possibilitando trocar a camada de interface de um serviço, mantendo a mesma lógica de aplicação. Nas próximas seções são descritas em mais detalhes a arquitetura de aplicação e arquitetura técnica da solução proposta.

4. A Arquitetura Baseada em Componentes

Um sistema colaborativo geralmente integra um conjunto de ferramentas para colaboração. Por exemplo, a maioria dos sistemas oferece fórum, bate-papo, agenda, relatórios de atividades, questionários, gerenciamento de tarefas, votação, repositório e links. Cada ferramenta é vista de forma relativamente independente dentro do sistema colaborativo. Estas características são propícias à aplicação de técnicas de desenvolvimento baseado em componentes, onde as ferramentas são componentes do sistema colaborativo a serem instanciados e configurados.

As ferramentas colaborativas possuem funcionalidades similares. Por exemplo, os serviços Conferências e Correio para Turma do ambiente AulaNet compartilham o suporte ao envio, ao recebimento e à exibição de mensagens, à categorização, à avaliação da participação e ao bloqueio do canal de comunicação, entre outras funcionalidades. Encapsular as funcionalidades recorrentes em componentes propicia também o reuso do suporte computacional à colaboração, aumentando o reuso de código. Passa também a ser possível evoluir, ajustar e construir serviços variando e reconfigurando os componentes de colaboração.

Sob esta ótica, o desenvolvimento de groupware é baseado em dois níveis de componentes. O primeiro nível contempla os componentes que provêem os serviços colaborativos, usados para oferecer suporte computacional à dinâmica da colaboração como um todo. O segundo nível contempla os componentes usados para montar ferramentas de colaboração, oferecendo suporte a determinados aspectos da colaboração dentro de uma ferramenta em particular. Nesta abordagem proposta, como ilustrado na Figura 2, os componentes que implementam as ferramentas colaborativas são chamados de serviços e os componentes usados para implementar o suporte computacional à colaboração dos serviços são chamados de componentes de colaboração. Mesmo um serviço de comunicação, como uma ferramenta de bate-papo, além dos componentes de

comunicação, também usa componentes de coordenação e de cooperação. Os componentes de colaboração de um C são reusados nos serviços dos demais C's.

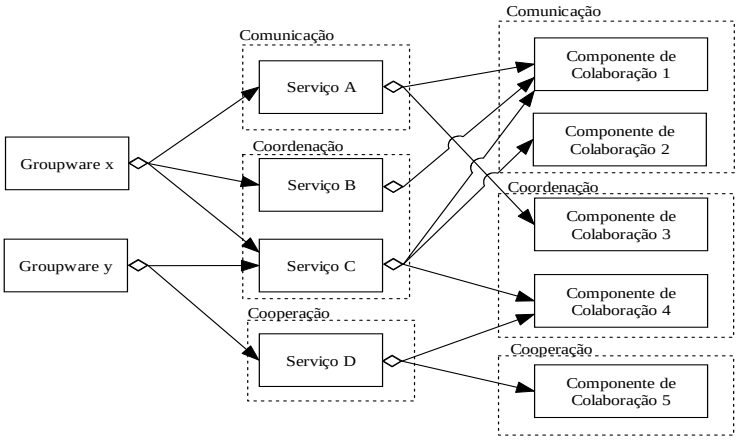

Figura 2. Sistemas colaborativos montados a partir de serviços, e serviços montados a partir de componentes de colaboração

O Modelo 3C de Colaboração é útil nesta abordagem para definir uma sistemática de classificação para os componentes. A partir de kits de componentes de colaboração, organizados em função do Modelo 3C, o desenvolvedor monta um serviço. Cada serviço, por sua vez também classificado em função do Modelo 3C, é usado para montar um sistema colaborativo. Além do reuso, esta abordagem favorece também a capacidade de extensão da solução ao possibilitar a inclusão de novos componentes.

Um serviço colaborativo normalmente possui suporte a funcionalidades referentes aos três Cs do modelo 3C. Um fórum de discussão, por exemplo, além dos componentes de comunicação, utiliza componentes de coordenação e de cooperação. As funcionalidades normalmente são recorrentes e relativamente autocontidas, como gerenciamento de sessão, permissão, avaliação, percepção, etc. Na abordagem proposta, estas funcionalidades são encapsuladas em componentes 3C.

O desenvolvedor de uma ferramenta colaborativa seleciona os componentes 3C que atendam às características do suporte à colaboração referentes à utilização da ferramenta no apoio à atividade colaborativa. Estes componentes encapsulam as complexidades técnicas e o suporte à colaboração e favorecem a concentração dos esforços do desenvolvedor na composição de um groupware específico. Os componentes encapsulam implementações e regras de negócio sobre colaboração, providas por especialistas do domínio e obtidas por experimentação, e reusadas em diversas situações.

A engenharia do domínio realizada para chegar ao conjunto de componentes foi conduzida seguindo o processo CBD-Arch-DE [Blois et al., 2004], com a adaptação de que a análise das ferramentas foi guiada pelo modelo 3C, sendo analisadas sob a perspectiva da comunicação, coordenação e cooperação. Foram identificadas as características recorrentes nos serviços colaborativos e chegou-se ao conjunto de componentes apresentado na Tabela 1.

COMUNICAÇÃO	COORDENAÇÃO	COOPERAÇÃO
MessageMgr	AssessmentMgr	CooperationObjMgr
TextualMediaMgr	RoleMgr	SearchMgr
VideoMediaMgr	PermissionMgr	VersionMgr
AudioMediaMgr	ParticipantMgr	StatisticalAnalysisMgr
PictorialMediaMgr	GroupMgr	RankingMgr
DiscreteChannelMgr	SessionMgr	RecommendationMgr
ContinuousChannelMgr	FloorControlMgr	LogMgr
MetaInformationMgr	TaskMgr	AccessRegistrationMgr
CategorizationMgr	AwarenessMgr	TrashBinMgr
DialogStructureMgr	CompetencyMgr	
ConversationPathsMgr	AvailabilityMgr	
CommitmentMgr	NotificationMgr	

Tabela 1. Componentes do Collaboration Component Kit

Um *component kit* não necessita ser exaustivo. Os *component kits* são extensíveis para acomodar novos componentes necessários. Componentes de software que sejam realmente reusáveis são refinados iterativamente até atingir a maturidade, a confiabilidade e a adaptabilidade desejadas [Gimenes & Huzita, 2005]. O conjunto de componentes é refinado a partir da realimentação obtida pelo reuso no desenvolvimento das aplicações e pela experimentação das diversas configurações no suporte às características dos grupos e tarefas envolvidos.

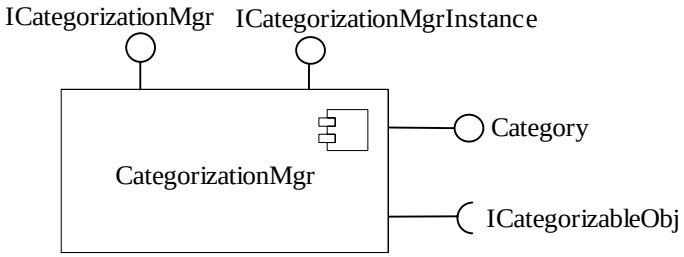

Figura 3. Componente de categorização de mensagens

Cada componente apresenta interfaces fornecidas e requeridas. A Figura 3 ilustra o componente de categorização de mensagens (CategorizationMgr). Este componente apresenta a interface ICategorizationMgr, que oferece os serviços relativos a categorização de mensagens em geral e ao ciclo de vida do componente; ICategorizationMgrInstance, que oferece os serviços referentes à utilização da categorização de mensagens, como atribuir categoria, modificar categoria, listar objetos de uma determinada categoria, etc.; Category, que representa a categoria em si; e ICategorizableObj, que é implementada no serviço colaborativo pelo respectivo objeto a ser categorizado.

4.1. Arquitetura Componentizada

Para oferecer suporte ao gerenciamento e à execução dos componentes, são usados *component frameworks* [Syzperski, 1997]. Um *component framework* é um conjunto de interfaces e regras de interação que possibilitam a implantação de componentes aderentes a um padrão. Na arquitetura proposta, é feito uso de um *component framework* para cada tipo de componente proposto. No Service Component Framework são acoplados os serviços, oferecendo suporte à montagem do sistema colaborativo, e no Collaboration Component Framework são acoplados os componentes de colaboração, usados na montagem do serviço. A Figura 4 ilustra a implantação dos componentes nos *component frameworks* correspondentes, que estabelecem as

condições ambientais para as instâncias dos componentes e oferecem serviços relativos ao ciclo de vida.

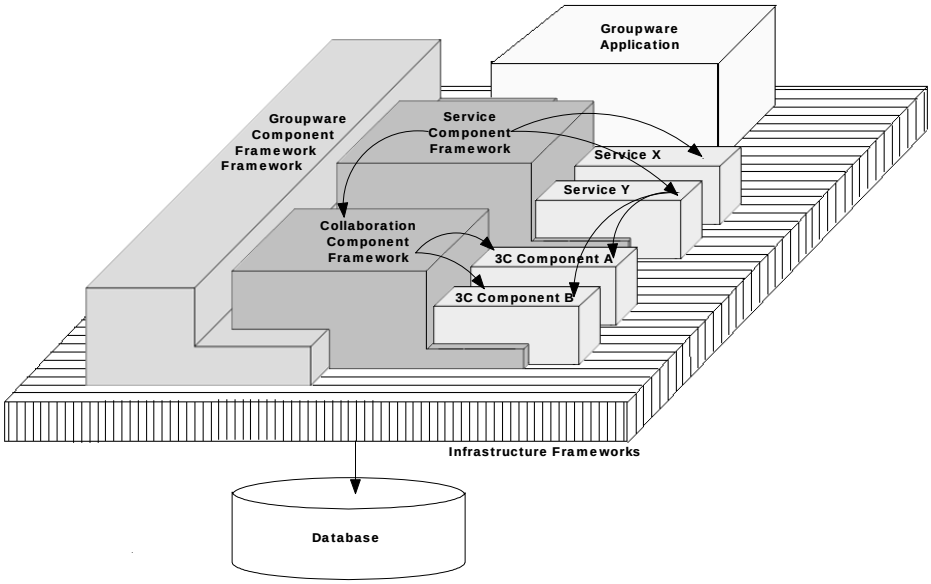

Figura 4. A arquitetura de aplicação proposta

Os *component frameworks* tratam a instalação, remoção, atualização, ativação, desativação, localização, configuração, monitoramento, importação e exportação de componentes. O Service Component Framework gerencia as instâncias dos serviços e a ligação com os componentes de colaboração correspondentes. O Collaboration Component Framework gerencia as instâncias dos componentes de colaboração, que são provenientes do Collaboration Component Kit.

Grande parte das funcionalidades dos *component frameworks* é recorrente e reusável. Na arquitetura proposta, é utilizado um framework para instanciar os *component frameworks*. Este tipo de framework é chamado de *component framework framework* (CFF) [Szyperski, 1997, p.277]. Um *component framework framework* é visto como um *component framework* de segunda ordem, onde seus componentes são *component frameworks* [Szyperski, 1997]. Da mesma forma que um componente interage com outros diretamente ou mediado pelo *component framework*, o mesmo pode ser dito dos *component frameworks*, cujo suporte de mais alto nível é o *component framework framework*. A Figura 4, estendendo a notação utilizada por Szyperski (1997), a arquitetura de aplicação é ilustrada, incluindo o Groupware Component Framework Framework, como o *component framework* de segunda ordem.

A arquitetura elaborada segue uma divisão em camadas. A camada de apresentação (não representada na Figura 4) é responsável pela captura e apresentação de dados e pela interação com o usuário; a camada de negócio captura o modelo da lógica de negócio do domínio da aplicação; e a camada de infra-estrutura implementa os serviços técnicos de baixo nível. A mesma infra-estrutura desenvolvida para a camada de negócio pode ser usada para mais de uma apresentação, como por exemplo, PDA, desktop implementado em HTML e desktop implementado em Flash (*Rich Internet Application*). Quando os serviços da camada de negócio necessitam de acesso remoto, como por exemplo, um cliente PDA, são disponibilizados web services que encapsulam a fachada da camada de negócio. Nos demais casos, a apresentação acessa diretamente a fachada do negócio.

O ferramental desenvolvido com esta pesquisa instrumenta o desenvolvimento da camada de negócio, implementando os conceitos do Modelo 3C de Colaboração. A arquitetura de aplicação expressa a estrutura dos componentes do domínio,

representando um projeto lógico de alto nível independente da tecnologia de suporte [D'Souza & Wills, 1998]. Instrumentado pelo Modelo 3C de Colaboração, o desenvolvedor modela a aplicação e seus requisitos, e seleciona os serviços e seus componentes de modo a oferecer suporte às necessidades de colaboração. O desenvolvedor seleciona os componentes desejados a partir dos *component kits*, implantando-os nos *component frameworks* correspondentes.

5. A Arquitetura Técnica do AulaNet

O AulaNet 3.0 é construído segundo uma arquitetura multicamadas que faz uso do padrão MVC [Fowler, 2002] e que integra frameworks de infra-estrutura [Fayad & Schmidt, 1997] [Fayad et al. 1999a][Fayad et al. 1999b]. A abordagem multicamadas em conjunto com o padrão MVC proporciona a separação entre a lógica da aplicação e a interface com o usuário, considerada uma boa prática de projeto de software [Fowler, 2002]. A Figura 5 esquematiza a arquitetura técnica do AulaNet 3.0.

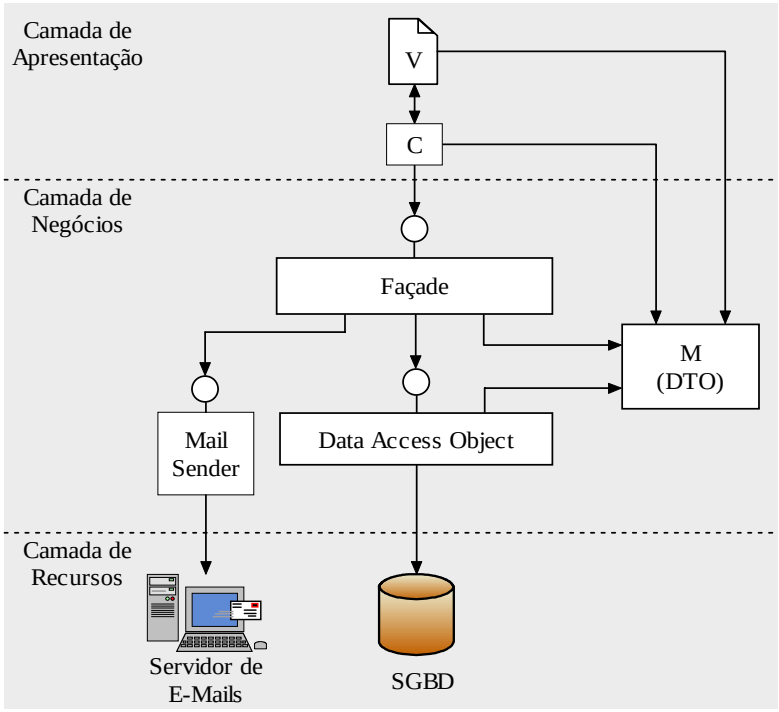

Figura 5. Arquitetura técnica do AulaNet 3.0 [Barreto, 2006]

As setas indicam o fluxo de controle da aplicação enquanto os retângulos representam classes e os círculos representam as interfaces. As linhas pontilhadas representam a divisão entre as camadas: recursos, negócios e apresentação. Essa arquitetura, definida em [Barreto, 2006], se baseia na Arquitetura de POJOs descrita por Johnson [2002, 2004].

A camada de recursos é composta de software externo a aplicação. Estão previstos inicialmente um servidor de e-mails e um Sistema Gerenciador de Bancos de Dados (SGBD), mas outros recursos podem ser acrescentados conforme demanda.

Na camada de negócios, o modelo (M do MVC) é implementado por classes que realizam o padrão de projetos *Data Transfer Object* (DTO) [Fowler, 2002], usadas para transportar os dados das entidades de negócio entre as diversas camadas. O acesso à base de dados é efetuado por classes que implementam o padrão de negócios *Data*

Access Object (DAO) [Alur et al., 2001]**,** que encapsula o acesso ao banco de dados e possibilita modificar a maneira de persistir as classes do modelo sem que seja preciso alterar o código cliente. *Mail Sender* é uma classe que encapsula o acesso ao servidor de E-Mails. A lógica de negócios é exposta para a camada de apresentação através do padrão *Façade* [Gamma et al., 1995], que provê uma interface para acesso às funcionalidades de um serviço.

A camada de apresentação expõe a lógica de negócios para o usuário-final. Essa camada é composta pelo controlador (C do MVC) e por páginas JSP representando a visão (V do MVC). O controlador invoca os métodos do *Façade* para acessar a camada de negócios. Os DTOs resultantes das operações são passados à visão que exibe as informações ao usuário.

Esta arquitetura é construída utilizando os frameworks de infra-estrutura Hibernate [Hibermate, 2007] e Spring [Spring, 2007] na camada de negócios e Spring MVC [Spring, 2007] na camada de apresentação. Esses frameworks visam tratar aspectos independentes de domínio, tais como persistência de dados, gerenciamento de transações, controle da interface gráfica etc. A incorporação destes frameworks possibilita que o desenvolvedor concentre-se nos aspectos funcionais da aplicação a partir de um nível mais alto de abstração.

O Hibernate, utilizado na camada de negócios, gerencia o mapeamento de classes em tabelas de bancos de dados, lidando com as questões decorrentes do conflito entre os paradigmas orientação a objetos e relacional [King & Bauer, 2005]. O Spring complementa o Hibernate adicionando serviços de controle de transações, segurança e exposição de serviços remotamente. Por fim, o Spring MVC é utilizado na camada de apresentação para gerenciar o fluxo de requisições entre visão e controlador, tratar e validar os parâmetros de requisições.

A arquitetura técnica do AulaNet 3.0 foi projetada de forma a possibilitar que o desenvolvedor de groupware concentre-se nos aspectos relevantes ao seu domínio enquanto questões de mais baixo nível são deixadas para frameworks de infra-estrutura. Além disso, a arquitetura do AulaNet 3.0 é expansível, possibilitando o acréscimo de outros frameworks [Barreto et al., 2006] ou o provimento de serviço para outros tipos de clientes [Barreto, 2006]. Na próxima seção é visto como é possível integrar o AulaNet 3.0 ao Ginga, possibilitando o uso desse ambiente na TV Digital.

6. A Integração do AulaNet com o Ginga

Ginga é o middleware que possibilita o desenvolvimento de aplicações interativas para o Sistema Brasileiro de TV Digital Terrestre (ISDTV-T). As aplicações para TV digital podem ser dividas em dois conjuntos de acordo com o tipo de conteúdo inicial: declarativas e procedimentais. Ginga provê suporte para ambas modalidades sendo a primeira através do módulo Ginga-NCL [Soares et al., 2007] e a segunda através do módulo Ginga-J [Souza et. al., 2007].

Linguagens declarativas provêem um maior nível de abstração para o desenvolvedor, porém por enfatizar um domínio restrito, são mais restritas do que as procedimentais. É possível a construção de aplicações híbridas em que os dois tipos de conteúdo coexistem e se referenciam [Souza et. al, 2007]. A arquitetura do AulaNet 3.0 é estendida para possibilitar a integração tanto com o Ginga-NCL quanto com o Ginga-J de forma a prover suporte aos dois tipos de aplicações para TV digital.

6.1. Integração com Ginga-NCL

Ginga especifica a linguagem NCL (*Nested Context Language*) para o desenvolvimento de aplicações declarativas. A linguagem é baseada em XML e tem como um dos seus principais objetivos definir como os diversos objetos de mídia são estruturados e relacionados no tempo e espaço. Para que o AulaNet seja acessível para usuários de TV digital, é preciso que além de páginas HTML, sejam providos documentos NCL.

Apesar da tecnologia JSP ser comumente usada para gerar conteúdo HTML, ela pode gerar também qualquer tipo de documento textual, inclusive scripts NCL. Desta forma, a arquitetura do AulaNet 3.0 é modificada acrescentando-se novos componentes na camada de apresentação para servir documentos NCL como pode ser visto na Figura 6.

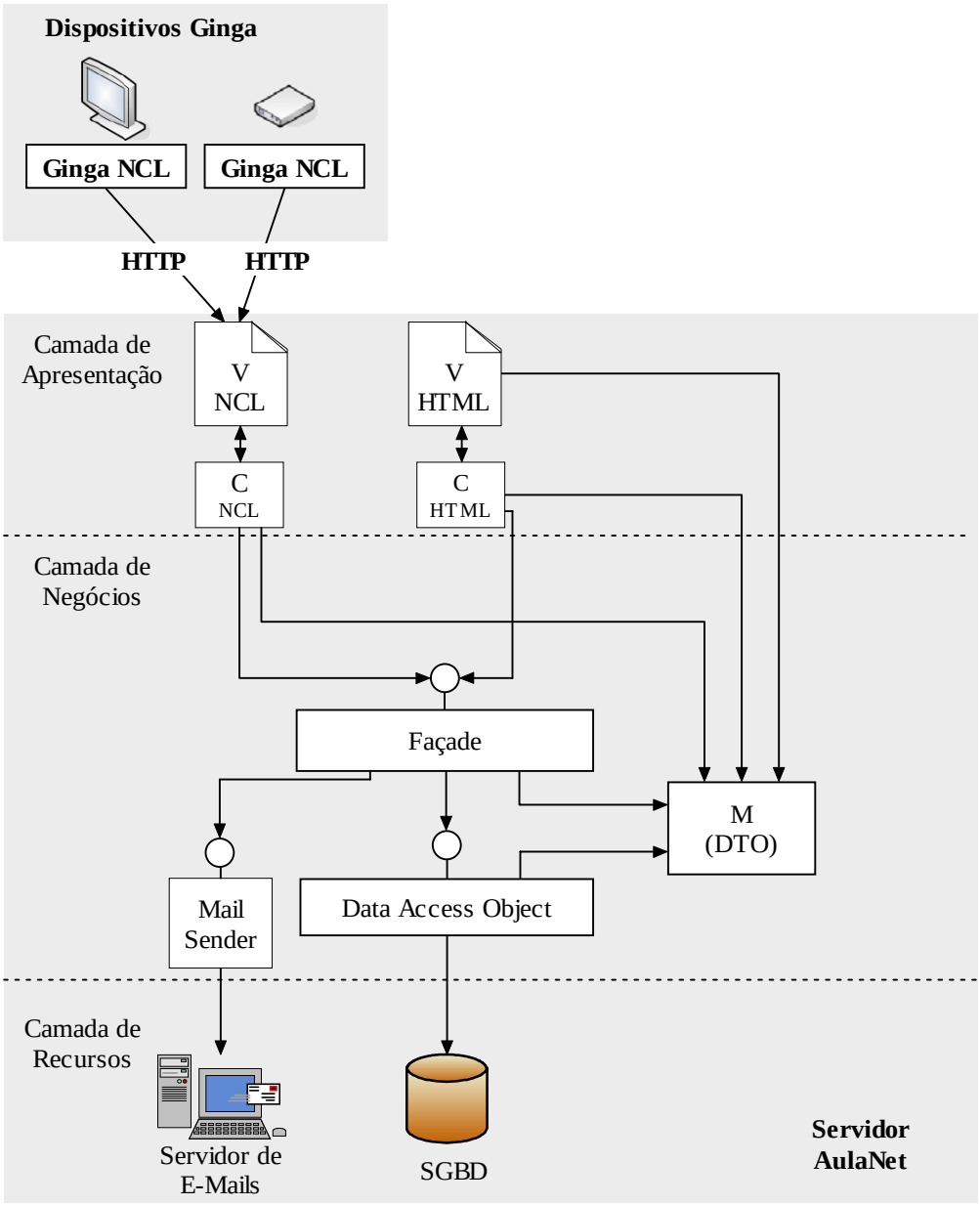

Figura 6. Arquitetura do AulaNet 3.0 com suporte ao Ginga-NCL

São adicionados novos elementos de Visão, identificados como V NCL no diagrama, que constituem os arquivos JSP que geram documentos NCL. Os JSPs que geram conteúdo HTML foram renomeados para V HTML. Da mesma forma, são adicionados

controladores para lidar com as requisições aos documentos NCL. Não há mudanças na camada de negócios e recursos, pois nesta arquitetura os usuários da TV Digital têm acesso aos mesmos serviços que os usuários da Web. Contudo, seria possível prover serviços exclusivos para a TV Digital através da adição de um novo *Façade*.

Os objetos de media referenciados no documento NCL são hospedados pelo AulaNet, e são retornados mediante requisição HTTP. Dessa forma, qualquer aparelho Ginga-NCL capaz de requisições HTTP pode interagir com o AulaNet 3.0.

6.2. Integração com Ginga-J

Ginga-J é o subsistema do middleware Ginga utilizado na execução de aplicações procedimentais para a TV digital. Essas aplicações, também conhecidas como Xlets, são executadas em uma máquina virtual Java, o que as torna independente de plataforma. Ginga-J especifica um conjunto de APIs para serem usadas no desenvolvimento de aplicações para a TV digital, incluindo as APIs da Sun JavaTV [Java TV, 2007], APIs de integração com dispositivos externos, APIs de envio de mensagens assíncronas dentre outras [Souza et al., 2007].

Não é viável ao AulaNet 3.0 gerar Xlets dinamicamente, como é feito com arquivos NCL. Torna-se necessária a implantação de uma infra-estrutura que possibilite a comunicação entre um Xlet em execução numa TV digital e os serviços do AulaNet. Em [Barreto, 2006] define-se uma camada de Web Services que expõe remotamente os serviços do AulaNet a dispositivos móveis. Podem ser providos serviços a Xlets de forma similar, como é esquematizado a seguir.

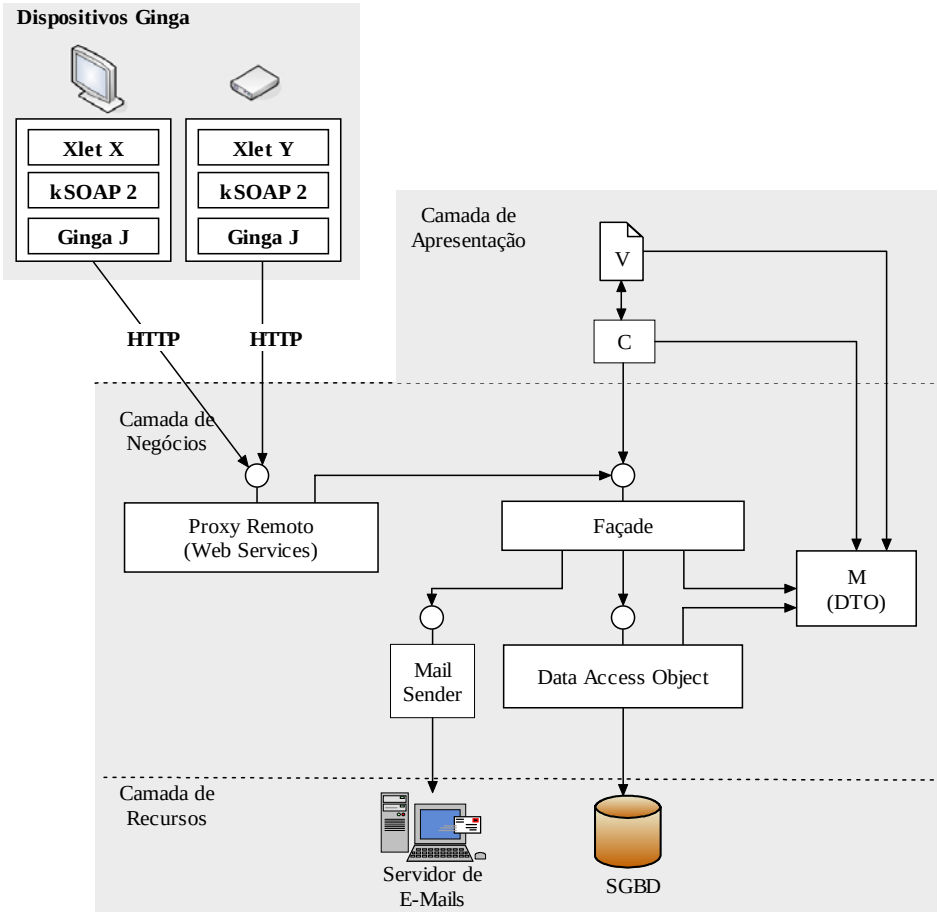

Figura 7. Arquitetura do AulaNet 3.0 com suporte ao Ginga-J

O Proxy remoto [Gamma et al., 1995] acrescentado expõe os serviços do AulaNet através de Web Services. Qualquer outro protocolo poderia ser utilizado, contudo escolheu-se Web Services por ser um padrão para integração entre sistemas heterogêneos [Singh et al., 2004].

O dispositivo Ginga contém um Xlet que utiliza a API kSOAP 2 [kSOAP, 2007], utilizada por ambientes com poucos recursos computacionais como por exemplo, aplicações J2ME ou JavaTV, para comunicação com Web Services.

Assim como no caso da integração com o Ginga-NCL, não há alteração nas visões e controladores que provêm serviços para os usuários de navegadores. Novos *Façades* podem ser acrescentados, provendo serviços específicos para usuários de TV digital. A comunicação é feita por HTTP, portanto é necessário que o dispositivo tenha acesso a internet.

7. Conclusão

"Sem uma arquitetura adequada, a construção de groupware e sistemas interativos em geral é difícil de obter, o software resultante é difícil de manter e o refinamento iterativo é dificultado" [Calvary et al., 1997]. Uma arquitetura componentizada possibilita que componentes sejam selecionados para montar um groupware voltado aos interesses específicos de um grupo. Os componentes são customizados e combinados na medida da necessidade, tendo em mente futuras manutenções. O uso desta abordagem propicia a

prototipação e a experimentação, que são fundamentais em sistemas colaborativos voltados para a TV digital, visto que ainda são escassos e pouco documentados os casos de sucesso. A prototipação deve ser rápida, pois ocorre enquanto o grupo trabalha, para se obter um feedback oportuno e proceder com os ajustes. O uso de componentes auxilia a adaptação dinâmica do ambiente e do suporte à colaboração através da recomposição e reconfiguração do sistema [Fuks et al., 2005].

Com alguns ajustes na arquitetura técnica do AulaNet 3.0 pode-se integrá-lo a um dispositivo Ginga que tenha acesso a internet, provendo meios de estender os cursos desse ambiente aos usuários de TV digital e abrindo novas possibilidades de ensino-aprendizagem colaborativo.

Ao integrar tanto com Ginga-J quanto com o Ginga-NCL, oferece-se ao desenvolvedor da aplicação de TV digital a possibilidade de trabalhar tanto com uma linguagem procedimental quanto com uma linguagem declarativa. Desta forma, pode-se utilizar o que for mais adequado para determinada situação. Também é importante ressaltar que a estrutura do AulaNet que serve clientes através de navegadores Web não sofre alteração e em tese o desenvolvimento habitual do AulaNet não sofre impacto devido ao acréscimo de serviços para a TV Digital.

Como trabalho futuro pretende-se implementar uma prova de conceito para a integração do AulaNet 3.0 com o Ginga. Também é preciso analisar questões não-funcionais tais como performance, escalabilidade e segurança. Por fim, devem-se explorar possibilidades de interações mais complexas do usuário da TV digital com o AulaNet.

Ao desenvolver um middleware de integração a partir da arquitetura baseada em componentes da nova versão do Ambiente AulaNet, o conhecimento e o acoplamento referente à tecnologia da TV digital ficará encapsulado nos componentes de software. Com isto, desenvolvedores podem focar seus esforços no projeto da interação e de dinâmica de colaboração em vez de ficarem presos em questões de baixo nível.

8. Bibliografia

Alur D., Crupi & J., Malks, D. (2001): Core J2EE Patterns: Best Practices and Design Strategies. Publisher: Prentice Hall / Sun Microsystems Press, 2001.

AMARAL, S.F. et al., Serviço de Apoio a Distância ao Professor em Sala de Aula pela TV Digital Interativa, Revista Digital de Biblioteconomia e Ciência da Informação, Campinas, V.1, N.2, ISSN 1678-765X, pp. 53-70, 2004.

Barreto (2006): Agregando Frameworks de Infra-Estrutura em uma Arquitetura Baseada em Componentes: Um Estudo de Caso no Ambiente AulaNet. Dissertação de Mestrado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), março de 2006.

Barreto, C. G., Filippo, D., Fuks, H. & Lucena, C. J. P. (2006): Integrating MAS in a component-based groupware environment. AOSE-2006@AAMAS Agent-Oriented Software Engeneering Workshop in AAMAS-International Joint Conference on Autonomous Agents & Multi-Agent Systems, Hakodate, Japão, 8 a 12 de maio de 2006, pp 145-156.

BELLONI, M.L. O que é Mídia e Educação. Campinas: Autores Associados, 2001.

Blois, A.P.T.B., Werner, C.M.L. & Becker, K. (2004) “Um Processo de Engenharia de Domínio com foco no Projeto Arquitetural Baseado em Componentes”, IV Workshop de Desenvolvimento Baseado em Componentes, João Pessoa, PB, pp. 15-20.

Borghoff, U.M. & Schlichter, J.H. (2000) Computer-Supported Cooperative Work: Introduction to Distributed Applications. Springer, USA.

Brown, S., Dalton, S., Jepp, D., Johnson, D., Li, S., Raible, M. (2005): Pro JSP, Fourth Edition, Appress, 2005.

Calvary, G., Coutaz, J. & Nigay, L. (1997) From Single-User Architectural Design to PAC*: a Generic Software Architectural Model for CSCW. Conference on Human Factors in Computing Systems (CHI’97), pp 242-249.

Councill, B. & Heineman, G.T. (2001) Definition of a Software Component and Its Elements, in: Component-Based Software Engineering: Putting the Pieces Together, Hineman, G.T. & Councill, W.T. (eds), Addison-Wesley, ISBN 0-201-70485-4.

CPqD, Política Regulatória: Panorama Brasileiro Atual – Projeto Sistema Brasileiro de Televisão Digital: Modelo de Implantação. Versão PD.30.12.36A.0002A/RT-05-AA. Campinas, CPqD, 2005, (Relatório Técnico, Cliente: Funttel, atividade 1236, OS: 40539).

D’SOUZA, D.F. & WILLS, A.C. Objects, Components and Frameworks with UML: The Catalysis Approach. Addison Wesley, ISBN 0-201-31012-0, 1998.

Edutools (2005) http://www.edutools.info (date 06/04/2005)

Ellis, C.A., Gibbs, S.J. & Rein, G.L. (1991) Groupware - Some Issues and Experiences. Communications of the ACM, Vol. 34, No. 1, pp. 38-58.

Fayad, M. E. & Schmidt, D. C. (1997): Object-oriented application frameworks, Communications of the ACM, v.40 n.10, p.32-38, Oct. 1997

Fayad, M. E., Schimidt & D. C., Johnson, R. E. (1999a): Implementing application frameworks: object-oriented frameworks at work. New York: J. Wiley, c1999. 729 p. ISBN 0471252018.

Fayad, M. E., Schimidt & D. C., Johnson, R. E. (1999b): Building application frameworks: object-oriented foundations of framework design. New York: J. Wiley, c1999. 664 p. ISBN 0471248754.

Fowler, M. (2002): Patterns of Enterprise Application Architecture. Addison-Wesley, 2002.

Fuks, H. (2000) “Aprendizagem e Trabalho Cooperativo no Ambiente AulaNet”, Revista Brasileira de Informática na Educação, N6, Abril 2000, ISSN 1414-5685, Sociedade Brasileira de Computação, pp 53-73, 2000.

Fuks, H., Raposo, A.B., Gerosa, M.A. & Lucena, C.J.P. (2005) Applying the 3C Model to Groupware Development. International Journal of Cooperative Information Systems (IJCIS), v.14, n.2-3, Jun-Sep 2005, World Scientific, ISSN 0218-8430, pp. 299-328.

Gamma, E., Helm, R., Johnson & R., Vlissides, J. (1995): Design patterns: elements of reusable object-oriented software. Addison-Wesley, Reading, MA, 1995.

GIMENES, I.M.S. & HUZITA, E.H.M. Desenvolvimento Baseado em Componentes, Editora Ciência Moderna, Rio de Janeiro, ISBN 85-7393-406-9, 2005.

Gutwin, C. & Greenberg, S. (2000) The Mechanics of Collaboration: Developing Low Cost Usability Evaluation Methods for Shared Workspaces. IEEE 9th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises -WETICE (2000), p. 98-103.

Hibernate (2007): Página oficial do framework Hibernate disponível em <http://www.hibernate.org/>. Última visita em 31/07/2007.

Hummes, J. & Merialdo, B. (2000) Design of Extensible Component-Based Groupware. Computer Supported Cooperative Work, 9(1), 53-74. ISSN 0925-9724.

JavaTV (2007): Sun Microsystems. JavaTV: Java Technology in Digital TV. Disponível em: <http://java.sun.com/products/javatv/>. Último acesso em 02/08/2007.

Johnson, R. (2002): Expert One-on-One J2EE Design and Development. Reading: Wiley Publishing Inc., 2002.

Johnson, R. (2004): Expert One-on-One J2EE Development without EJB. Wiley Publishing Inc., 2004.

King, G. & Bauer, C. (2005): Hibernate in Action. Manning Publishing Co., 2005.

Krueger, C.W. (1992) Software Reuse, ACM Computing Surveys, Volume 4 Issue 2 p131-183.

kSOAP (2007): Página oficial do kSOAP 2 disponível em < http://ksoap2.sourceforge.net/ >. Última visita em 02/08/2007.

Li, D. & Muntz, R.R. (1998) "COCA: Collaborative Objects Coordination Architecture", Proceedings of the ACM Conference on Computer Supported Cooperative Work 1998, pp. 179-188.

Moore, J. M. & Bailin, S.C. (1991) "Domain Analysis: Framework For Reuse", Prieto-Diaz, R. & Arango, G. (eds.), Domain Analysis and Software System Modeling. LosAlamitos, CA: IEEE Computer Society Press, pp. 179-203.

OLIVEIRA, E.C.R., ALBUQUERQUE, C.V.N. TV Interativa: Uma Alternativa para o Processo de Aprendizagem. In: World Congress on Computer Science, Engineering and Technology Education, 2006, Itanhaém. WCCSETE´06, 2006

Pfleeger, S.L. (2001) Software engineering: theory and practice, Upper Saddle River, NJ: Prentice Hall.

SANCRINI, M. O Uso da Televisão Digital no Contexto Educativo, Educação Temática Digital, Campinas, V 7, N 1, ISSN 1676-2592, pp. 31-44, 2005.

SANTOS, D. T., SILVA, M.R.C, MELONI, L.G.P. Ferramentas de Apoio ao Ensino a Distância via TV Digital Interativa. In: Taller Internacional de Software Educativo, Santiago-Chile. 2005.

Singh, I., Stearns, B., Brydon, S., Murray, G. & Ramachandran, V. (2004): Designing Web Services with J2EE 1.4 Plataform. Pearson Education, 2004.

Slagter, R.J. & Biemans, M.C.M. (2000) "Component Groupware: A Basis for Tailorable Solutions that Can Evolve with the Supported Task", in Proceedings of the International ICSC Conference on Intelligent Systems and Applications (ISA 2000), Wollongong, Australia.

Soares, L. F. G., Rodrigues, R. F., Moreno, M. F. (2007): Ginga-NCL: the Declarative Environment of the Brazilian Digital TV System. Journal of the Brazilian Computer Society, Revista no. 4; Vol. 12; Mar. 2007 - ISSN 0104-6500.

Souza, G. L. F., L. E. C. Leite, Batista, C. E. C. F. (2007): Ginga-J: The Procedural Middleware for the Brazilian Digital TV System. Journal of the Brazilian Computer Society, Revista no. 4; Vol. 12; Mar. 2007 - ISSN 0104-6500.

Spring (2007): Página oficial do framework Spring disponível em <http://www.springframework.org/>. Última visita em 31/07/2007.

SZYPERSKI, C. Component Software: Beyond Object-Oriented Programming, Addison-Wesley, ISBN 0-201-17888-5, 1997.

Tam, J. & Greenberg, S. (2004) A framework for asynchronous change awareness in collaboratively-constructed documents, CRIWG 2004, LNCS 3198, p. 67-83.

Teles, V.M. (2004) Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade, ed. Novatec, ISBN 8575220470.

TOME, T.; PESSOA, A.C.F.; RIOS, J.M.M.; LOURAL, C.A.; DALL'ANTONIA, J.C., Relatório Integrador dos Aspectos Técnicos e Mercadológicos da Televisão Digital. Versão AB PD.33.SV.E5A.0005A/RT-02-AB. Campinas: CPqD, 2001, 84p.

WAISMAN, T. TV Digital Interativa na educação: afinal, interatividade para quê? Escola do Futuro da USP, 2002. Disponível em: <www.futuro.usp.br/producao_cientifica/artigos/itv.pdf>. 10/06/2007.

Won, M., Stiemerling, O. & Wulf, V. (2005) "Component-Based Approaches to Tailorable Systems", End User Development, Lieberman, H.;, Paterno, F. & Wulf, V. (eds), Kluwer, pp. 1-27.

9. Biografia Simplificada dos Autores

Hugo Fuks é Professor Associado do Departamento de Informática da Pontificia Universidade Católica do Rio de Janeiro. Obteve o grau de Doutor em Ciência da Computação pelo Imperial College da Universidade de Londres em 1991. Outros graus acadêmicos são Mestre em Informática, PUC-Rio, 1984; Engenheiro de Produção, UFRJ, 1981 e Analista de Sistemas, CCE-PUC/Rio, 1981.

Marco Aurélio Gerosa é pesquisador do Laboratório de Engenharia de Software da Pontificia Universidade Católica do Rio de Janeiro (PUC-Rio), onde obteve o título de doutor e mestre em Informática. Sua tese de doutorado foi sobre desenvolvimento de groupware componentizado com base no modelo 3C de colaboração. Atualmente é professor titular do Centro Universitário Vila Velha (UVV). É formado em Engenharia de Computação pela Universidade Federal do Espírito Santo (UFES).

Celso Gomes Barreto é pesquisador do Laboratório de Engenharia de Software da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio). Obteve o título de mestre em Informática em 2006 após defender a dissertação "Agregando Frameworks de Infra-Estrutura em uma Arquitetura Baseada em Componentes: Um Estudo de Caso no Ambiente AulaNet". É formado em Ciência da Computação pela Universidade Federal do Rio de Janeiro (UFRJ).

Carlos José Pereira de Lucena é professor de Engenharia de Software do Departamento de Informática da PUC-Rio, onde coordena o Laboratório de Engenharia de Software. É também Adjunct Professor do Computer Science and Senior Research Associate, Computer Systems Group, University of Waterloo. Lucena recebeu seu título de doutorado na Universidade da Califórnia, em Los Angeles.